

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ по
лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Рекурсия

Студент гр. 8304

Самакаев Д.И.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2019

Вариант 8

Цель работы.

Изучить основы рекурсивных функций и эффективных алгоритмов.

Постановка задачи.

1. Разработать программу, использующую рекурсию;
2. Сопоставить рекурсивное решение с итеративным решением задачи;
3. Сделать вывод о целесообразности и эффективности рекурсивного решения данной задачи.

Построить синтаксический анализатор для простое_логическое.
простое_логическое ::= true | false | простой_идентификатор |
NOT простое_логическое | (простое_логическое знак_операции
простое_логическое) простой-идентификатор ::= буква
знак-операции ::= AND | OR

Описание алгоритма.

На каждом шаге алгоритма проверяемое выражение делится на части до тех пор, пока не сведётся к простейшему виду, указанному в условии задачи.

Спецификация программы.

Программа предназначена для синтаксического анализа выражения методом рекурсии.

Программа написана на языке C++. Входные данные подаются в виде строк текстового файла или консольным вводом.

Описание функций.

1. int is_operand(string s)

Определяет, начинается ли строка с оператора и возвращает количество символов в этом операторе.

2. `int operand_pos_search(string s)`

Выполняет поиск оператора в строке и возвращает индекс его первого символа в строке.

3. `bool is_prime_logical(string s)`

Рекурсивная функция, синтаксический анализатор выражения.

Вывод.

Был получен опыт работы с рекурсией и синтаксическим анализатором. Была реализована программа для выявления простых логических выражений.

Приложение.

1)Тестирование.

```
<(a OR b) AND (NOT c AND d)> is prime logical
(a b NOT a b c) is not prime logical
a AND b is not prime logical
a is prime logical
TRUE is prime logical
(TRUE OR FALSE) is prime logical
(a OR TRUE) is prime logical
NOT a is prime logical
NOT (a OR b) is prime logical
NOT a OR a is not prime logical
b OR b OR a is not prime logical
```

((a OR b) AND (NOT c AND d))	является простым логическим
(a b NOT a b c)	не является простым логическим
a AND b	не является простым логическим
a	является простым логическим
TRUE	является простым логическим
(TRUE OR FALSE)	является простым логическим
(a OR TRUE)	является простым логическим
NOT a	является простым логическим
NOT (a OR b)	является простым логическим
NOT a OR a	не является простым логическим
b OR b OR a	не является простым логическим

2)Исходный код.

```
#include <iostream>
#include <string>
#include <fstream>

//является ли начало строки оператором
int is_operand(std::string s)
{
    if (!s.compare(0, 5, " AND "))
        return 5;
    else if (!s.compare(0, 4, " OR "))
        return 4;
    else return 0;
}
//находит позицию оператора AND или OR
int operand_pos_search(std::string s)
{
    int len = s.length();
    std::string side_str;
```

```

int n = 0;
for (int i = 0; i < len; i++)
{
    side_str = s;
    if (s[i] == '(')
        n++;
    else if (s[i] == ')')
        n--;
    if (n == 0 && is_operand(side_str.erase(0, i + 1)))
        return i + 1;
}
return -1;
}

//основная функция
bool is_prime_logical(std::string s)
{
    std::string side_str = s;
    int len = s.length();
    if ((isalpha(s[0]) && len == 1) ||
        (s == "TRUE") ||
        (s == "FALSE"))
        return true;
    else if (!(s.compare(0, 4, "NOT ")))
    {
        return is_prime_logical(side_str.erase(0, 4));
    }
    else if (s[0] == '(' && s[len - 1] == ')')
    {
        s.erase(len - 1, 1);
        s.erase(0, 1);
        side_str = s;
        len -= 2;
        int operand_pos = operand_pos_search(s);
        if (operand_pos == -1)
            return false;
        side_str.erase(operand_pos);
        if (is_prime_logical(side_str))
        {
            side_str = s;
            int operand_len = is_operand(side_str.erase(0, operand_pos));
            side_str = s;
            if (is_prime_logical(side_str.erase(0, operand_len + operand_pos)))
                return true;
            else return false;
        }
        else return false;
    }
    else return false;
}

void file_input(char* argv) {
    std::ifstream file;
    std::string testfile = argv;
    file.open(testfile);
    if (!file.is_open())
        std::cout << "Error! File isn't open" << std::endl;
    std::string str;
    while (!file.eof()) {
        getline(file, str);
        if (is_prime_logical(str))
            std::cout << str << " is prime logical" << std::endl;
        else std::cout << str << " is not prime logical" << std::endl;
    }
}

```

```

        file.close();
    }

    void console_input()
    {
        std::string s;
        std::cout << "Enter the pattern to check" << std::endl;
        std::cin >> s;
        if (is_prime_logical(s))
            std::cout << s << "    is prime logical" << std::endl;
        else std::cout << s << "    is not prime logical" << std::endl;
    }

    //аргументом передаётся имя тестового файла
    int main(int argc, char** argv)
    {
        if (argc == 1)
            console_input();
        else if (argc == 2)
            file_input(argv[1]);
        else
            std::cout << "too much arguments" << std::endl;

        return 0;
    }

```